Micro Focus Visual COBOL チュートリアル

IIS と .NET COBOL コンポーネントを連携させた RESTful Web サービスの開発

1. 目的

Visual COBOL では、他言語・他システムとの連携手法として、「Enterprise Server」を利用したサービス連携だけではなく、COBOL プログラムに一切の変更を行わずに、 .NET 技術と連携可能な .NET COBOL 機能も提供しています。本機能を利用することで、C#、VB .NET などの .NET 言語で作成した RESTful Web サービス上で COBOL を利用するといった方法も可能となります。

このドキュメントでは .NET COBOL 機能を利用して C# で実装する RESTful Web サービスと COBOL の連携方法について説明します。

2. 前提条件

本チュートリアルは、下記の環境を前提に作成されています。

- 開発クライアント ソフトウェア

OS	Windows 10 Enterprise Edition (64bit)
COBOL 開発環境製品	Micro Focus Visual COBOL 7.0J for Visual Studio

本資料では Visual Studio 2019 を使用しています。Visual Studio のバージョンにより、設定画面への遷移方法やレイアウトなどが異なる場合があります。

また、RESTful Web サービスの実装やテストクライアントなどにおいて、C#、IIS サーバー、jQuery などの技術を利用したチュートリアルとなっておりますが、これらの技術に関する説明は本チュートリアルでは行っておりません。別途資料やオンライン文献などを参照ください。

- チュートリアル用サンプルプログラム

下記のリンクから事前にチュートリアル用のサンプルファイルをダウンロードして、任意のフォルダに解凍してください。

このサンプルプログラムは、COBOL で作成された簡単な書籍情報を管理するアプリケーションであり、索引ファイルを利用しています。

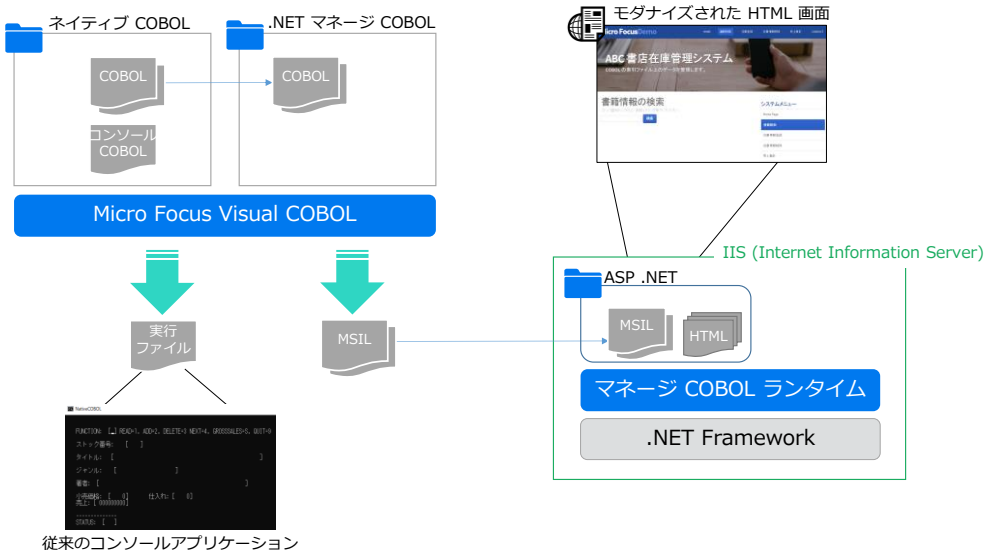
サンプルプログラムのダウンロード

内容

3. チュートリアル手順の概要

3.1. 今回作成するアプリケーション概要

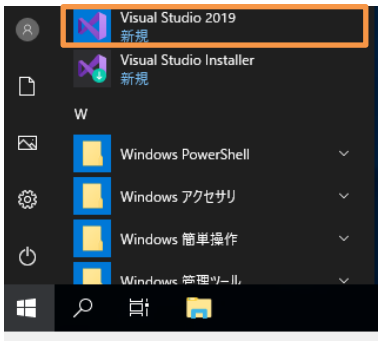
従来のコンソールアプリケーションである書籍情報を管理するアプリケーションを、.NET COBOL の機能を利用して COBOL プログラムを変更することなく MSIL コードを生成します。生成された MSIL を基に、Windows の IIS サーバーに RESTful Web アプリケーションとして登録した後、登録したアプリケーションが実際に動作することを確認します。



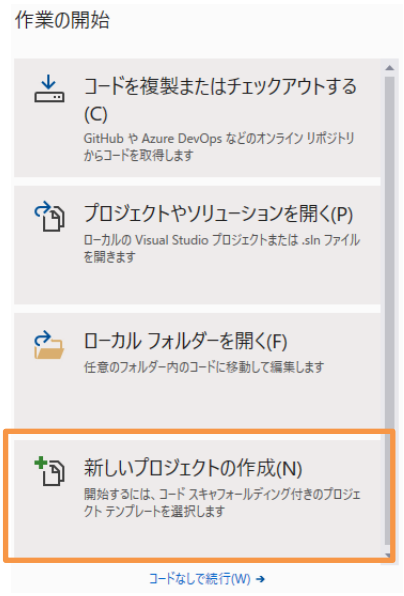
3.2. プロジェクトの作成と COBOL アプリケーションの確認

3.2.1. プロジェクトの作成

1) Visual Studio XXXX (XXXX はバージョン番号) を起動します。



2) [新しいプロジェクトの作成(N)] をクリックします。



3) 以下のフィルタを設定し、[コンソールアプリケーション] を選択し、[次へ(N)] をクリックします。

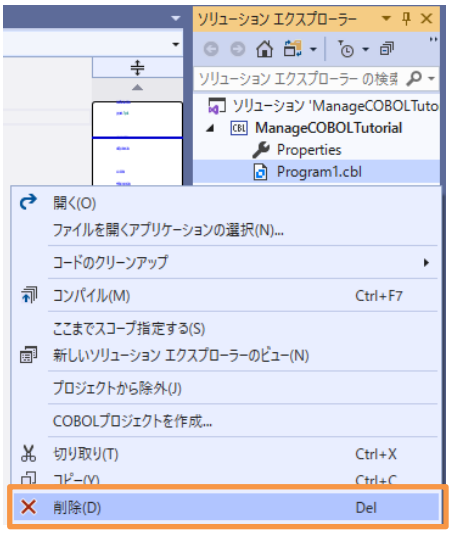
全ての言語： COBOL

全てのプラットフォーム： Windows

全てのプロジェクトの種類： コンソール

4) 「プロジェクト名」に “ManageCOBOLTutorial” を入力し、[作成(C)] をクリックします。

5) 自動作成された Program1.cbl は不要のため、ソリューションエクスプローラーより、「Program1.cbl」を選択し、マウスの右クリックからコンテクストメニューを表示した上で、[削除(D)] を選択します。

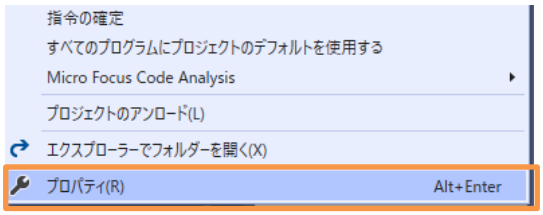


6) チュートリアルファイル解凍フォルダ¥COBOL フォルダ配下の、BOOKSCRN.cbl, BOOK.cbl, BOOK-INFO.cpy をプロジェクト内にドラッグアンドドロップします。

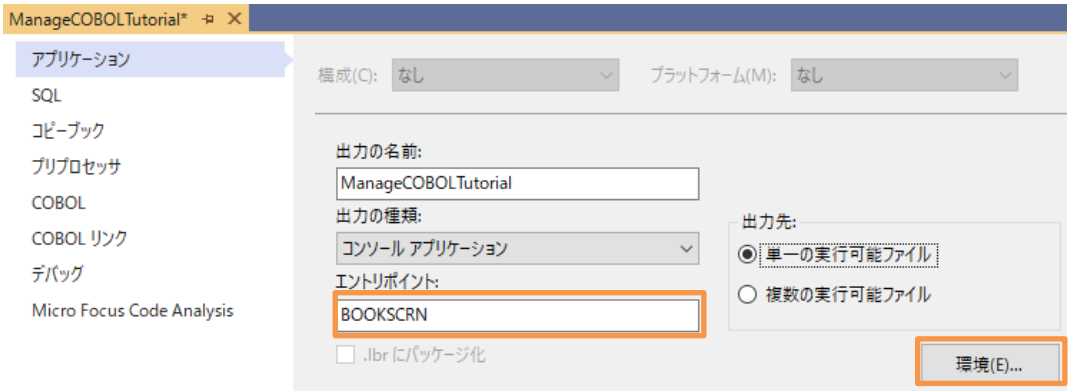
7) チュートリアルファイル解凍フォルダ配下の DAT フォルダを任意のフォルダにコピーします。フォルダ配下には、書籍情報を管理する索引ファイルが保存されています。なお、本チュートリアルでは、C:¥ 直下としています。

3.2.2. アプリケーションの動作確認

1) ManageCOBOLTutorial プロジェクト名を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[プロパティ(R)] を選択します。



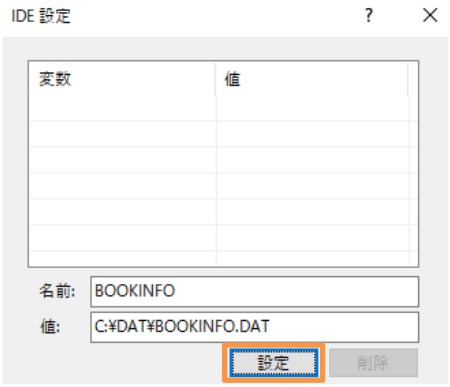
2) 「アプリケーション」タブを選択し、「エントリポイント」に “BOOKSCRN” を入力した上で、[環境(E)] をクリックします。



3) 以下の入力を行い、[設定] をクリックします。

名前： “BOOKINFO”

値： “C:¥DAT¥BOOKINFO.DAT”

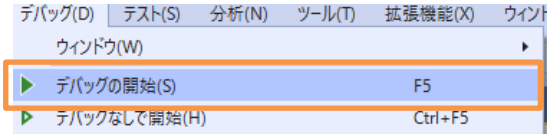


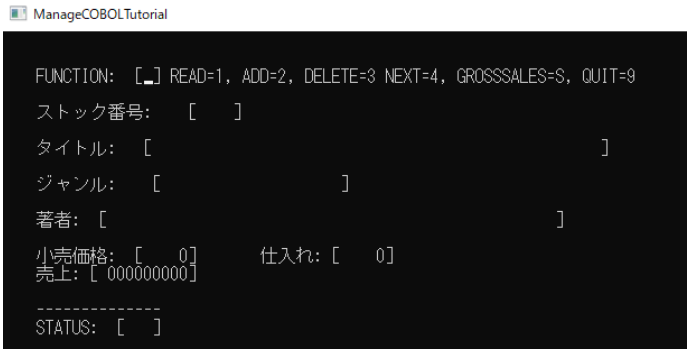
4) 追加されたことを確認の上、[OK(O)] をクリックします。

5) 「COBOL」タブを選択し、「追加指令」欄に “ASSIGN(EXTERNAL)” を入力します。

6) [ファイル(F)] > [すべて保存(L)] で変更を保存します。

7) [デバッグ(D)] > [デバッグの開始(S)] をクリックします。



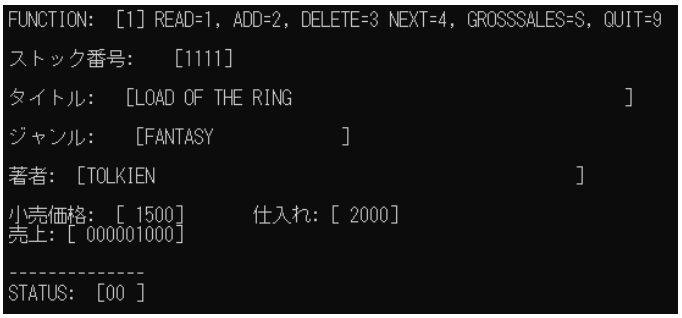


上記のような画面が表示されます。

以下の入力を行った後、Enter キーを打鍵すると、書籍情報が取得できます。

FUNCTION: “1”

ストック番号 : “1111”



現在、ストック番号 : 4444 の書籍は未登録のため、以下の情報を入力し、Enter キーを打鍵することで、情報を追加します。

FUNCTION: “2”

ストック番号： “4444”

タイトル： “ブレイブストーリー”

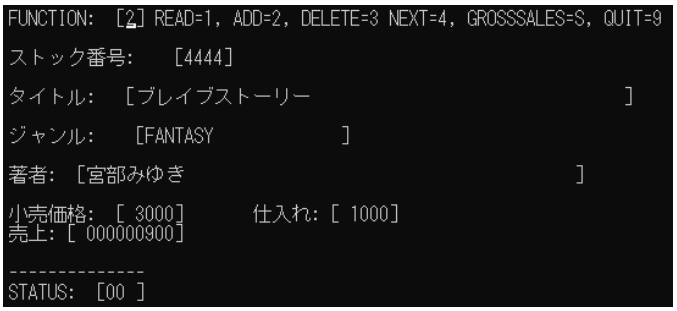
ジャンル： “FANTASY”

著者： “宮部みゆき”

小売価格： “3000”

仕入れ： “1000”

売上： “900”

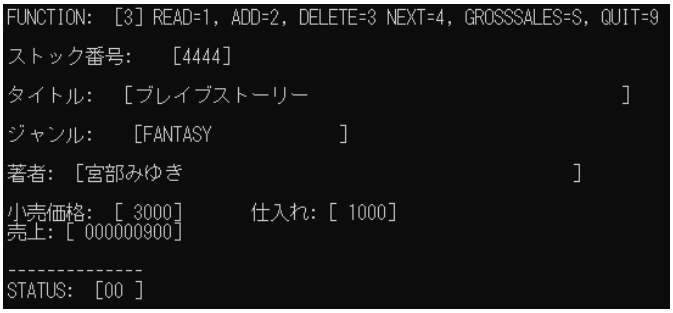


実行後、READ 機能を用いて、追加した情報が参照できていることを確認してください。

確認した後、以下の入力後、Enter キーを打鍵することで、情報を削除します。

FUNCTION: “3”

ストック番号： “4444”

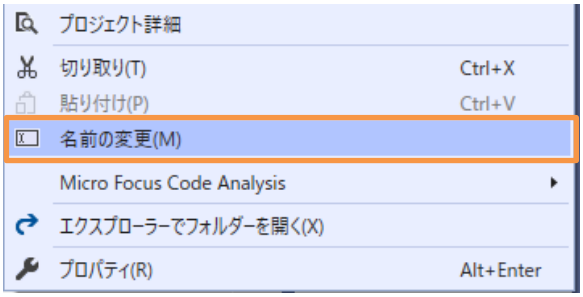


実行後、READ 機能でストック番号： 4444 が削除されていることを確認してください。

FUNCTION=S の GROSSSALES 機能は登録された全書籍情報の売上額を集計します。

確認後、FUNCTION=9 でアプリケーションを終了してください。

8) 後の手順で新たなプロジェクトを作成するため、今回のプロジェクト名を変更します。ソリューションエクスプローラーより、ManageCOBOLTutorial プロジェクト名を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[名前の変更(M)] をクリックし、”NativeCOBOL” とプロジェクト名を変更します。



3.3. .NET COBOL プロジェクトの作成

本節では、さきほど確認した従来の COBOL プログラムを .NET COBOL としてコンパイルを行います。

1) Visual Studio IDE のソリューションエクスプローラーより、”ManageCOBOLTutorial” ソリューション名を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[追加(D)] > [新しいプロジェクト(N)] をクリックします。

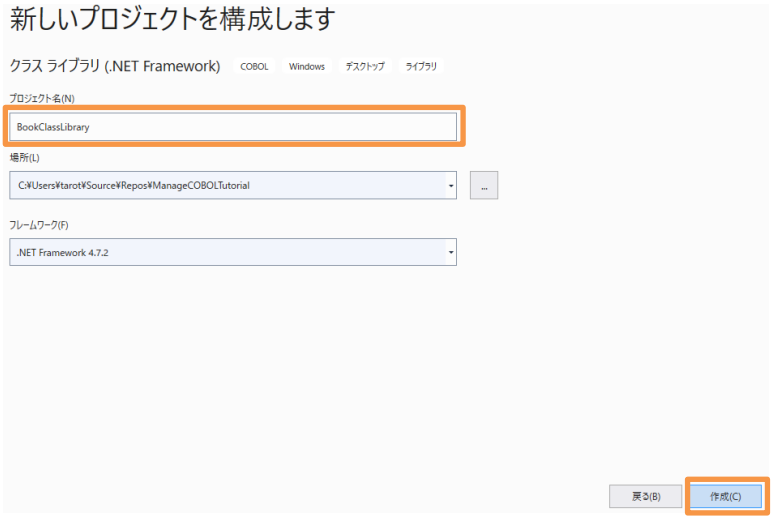
2) 以下のフィルタを設定し、[クラスライブラリ (.NET Framework)] を選択し、[次へ(N)] をクリックします。

全ての言語： COBOL

全てのプラットフォーム： Windows

全てのプロジェクトの種類： ライブラリ

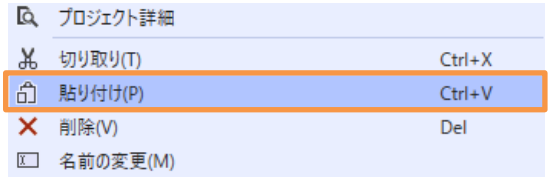
3) プロジェクト名に “BookClassLibrary” を入力し、[作成(C)] をクリックします。



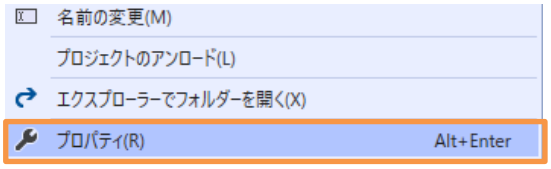
4) BookClassLibrary プロジェクト作成時に自動作成される Class1.cbl は不要なため、削除してください。

5) NativeCOBOL プロジェクト配下の BOOK.cbl, BOOK-INFO.cpy を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[コピー(Y)] をクリックします。

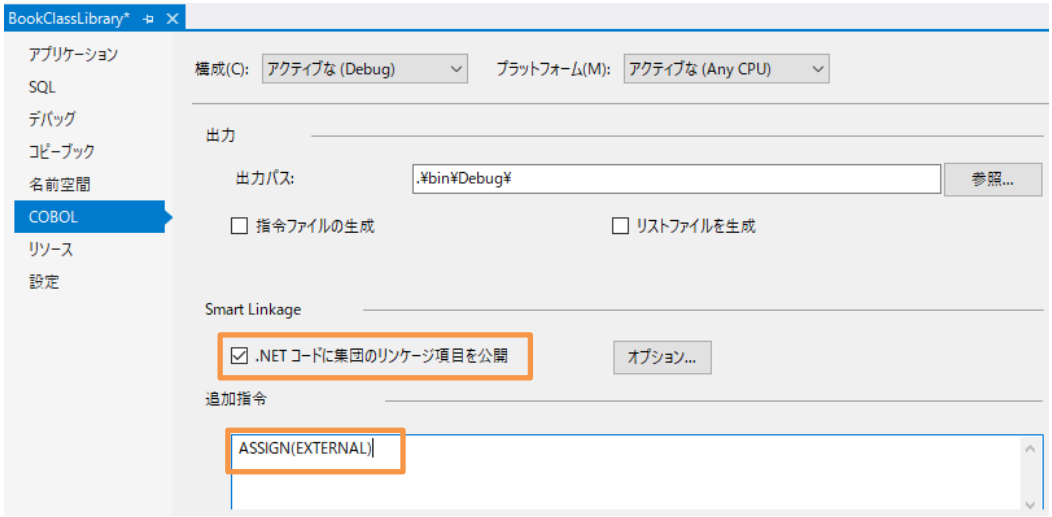
6) BookClassLibrary プロジェクト名を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[貼り付け(P)] をクリックします。



7) BookClassLibrary プロジェクト名を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[プロパティ(R)] をクリックします。

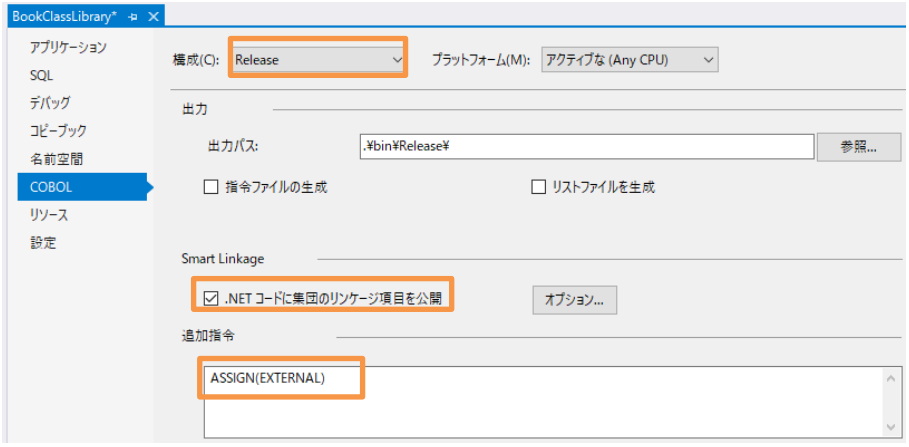


8) 「COBOL」タブを選択し、[.NET コードに集団のリンケージ項目を公開] にチェックを行い、追加指令に “ASSIGN(EXTERNAL)” を入力します。

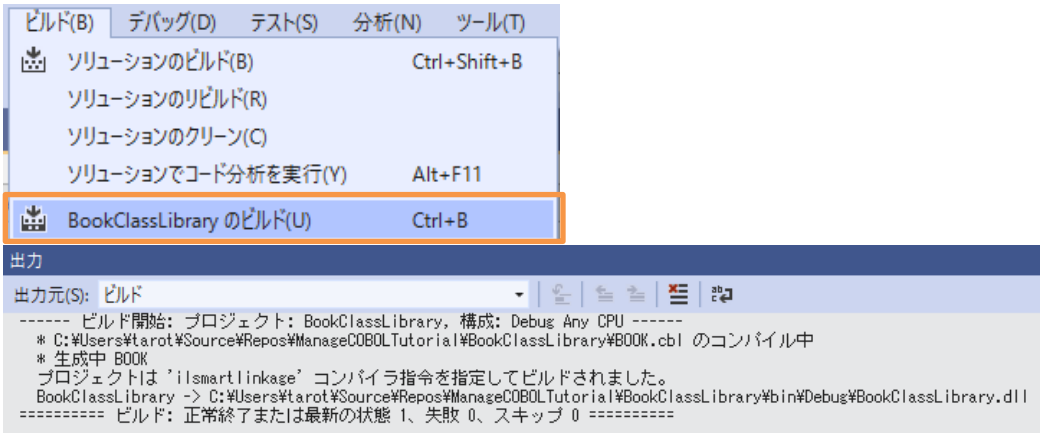


補足）
[.NET コードに集団のリンケージ項目を公開] にチェックを行うことで、ILSMARTLINKAGE というコンパイラ指令を設定したことになります。
本指令の詳細については、製品マニュアルトップより、[リファレンス] > [コンパイラ指令] > [.NET COBOL コマンド ライン コンパイラ指令] > [コード生成指令] > [ILSMARTLINKAGE] を参照してください。

9) 構成を “Release” に変更し、前手順と同様の設定を行い、保存します。



10) [ビルド(B)] > [BookClassLibrary のビルド(U)] を選択して、ライブラリのビルドを行います。

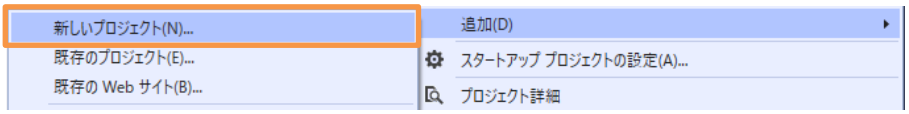


生成された BookClassLibrary.dll は Visual COBOL のコンパイラにより、コードの変更なく .NET Framework 上で動作するよう、MSIL 形式で DLL を作成しています。このため、C#, VB .NET のような .NET 言語と連携することが可能になります。

3.4. C# による RESTful Web サービスの作成

さきほど作成した .NET COBOL のライブラリを RESTful Web サービスから呼び出します。RESTful Web サービスは、C# で作成します。

1) ManageCOBOLTutorial ソリューション名を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[追加(D)] > [新しいプロジェクト(N)] を選択します。

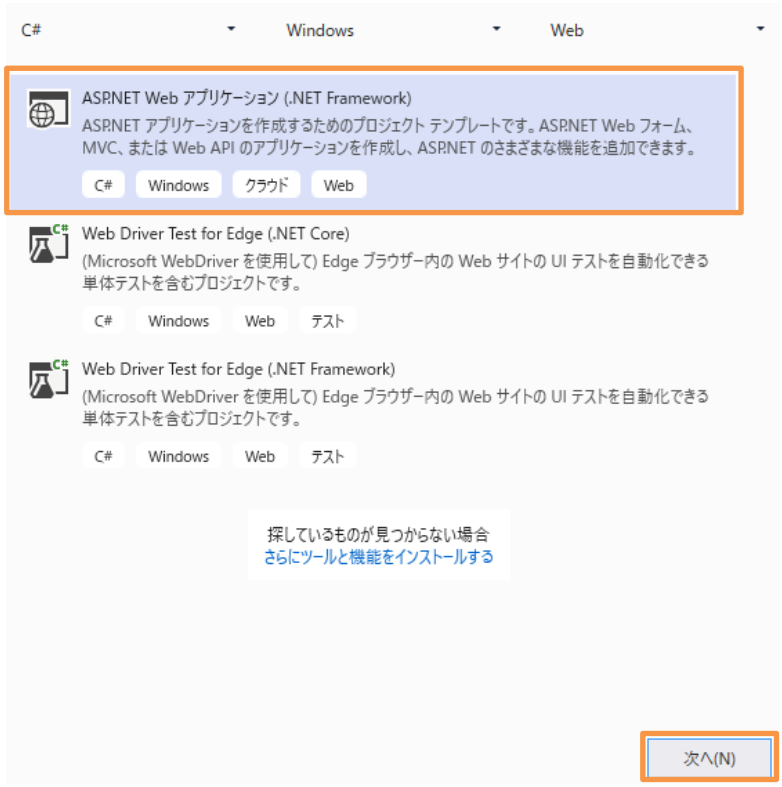


2) 以下のフィルタを設定し、[ASP.NET Web アプリケーション (.NET Framework)] を選択し、[次へ(N)] をクリックします

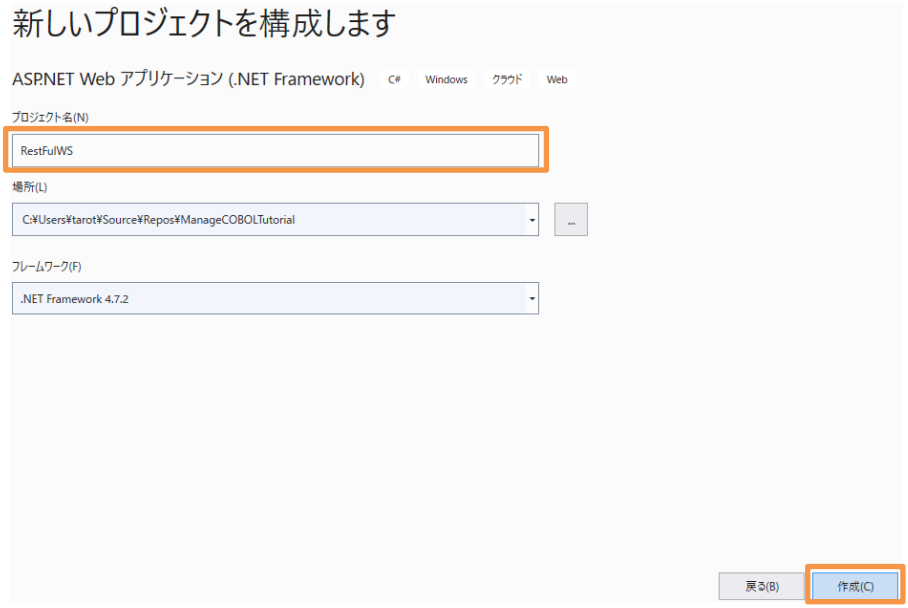
全ての言語： C#

全てのプラットフォーム： Windows

全てのプロジェクトの種類： Web

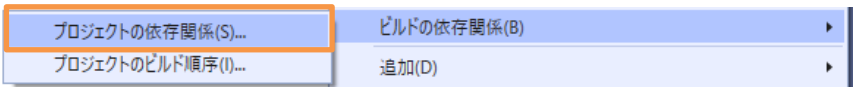


3) プロジェクト名に “RESTfulWS” を入力し、[作成(C)] をクリックします。

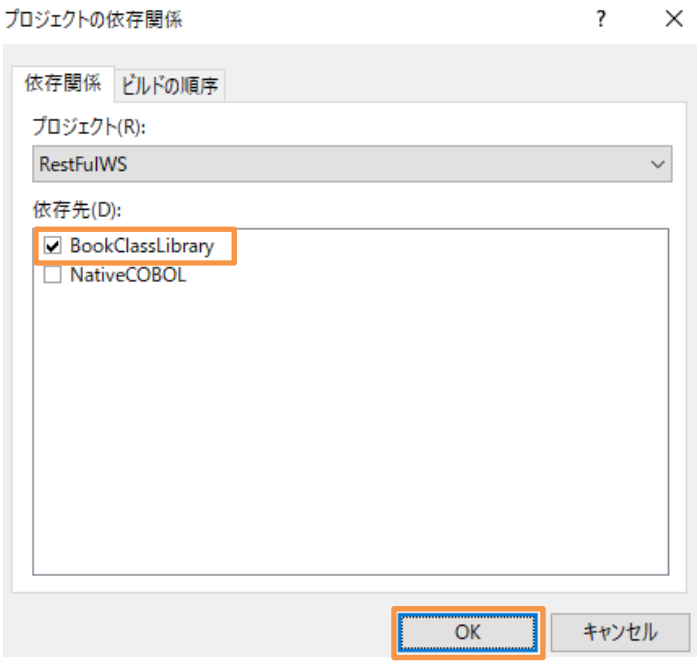


4) 「Web API」を選択し、[作成] をクリックします。

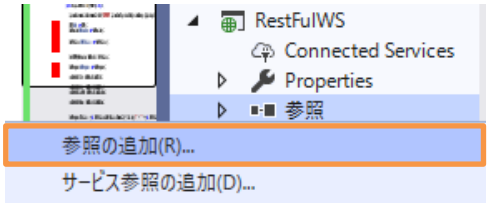
5) RESTfulWS プロジェクト名を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[ビルドの依存関係(B)] > [プロジェクトの依存関係(S)] を選択します。



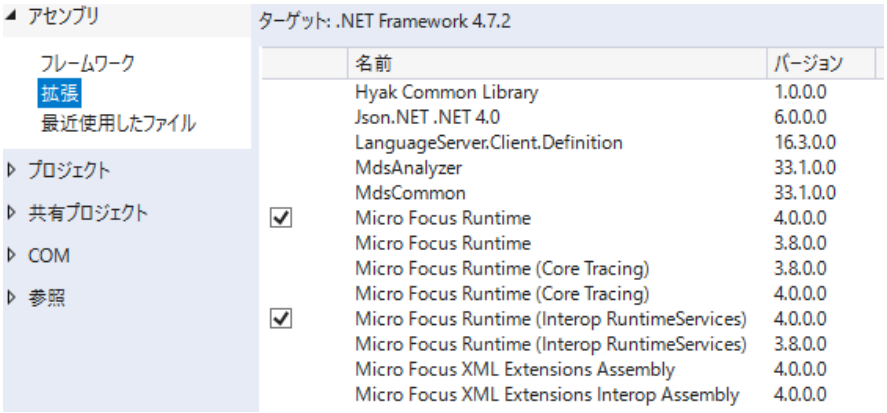
6) 「依存関係」 タブより、「BookClassLibrary」 にチェックを行った後、[OK] をクリックします。



7) RESTfulWS プロジェクト配下の参照を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[参照の追加(R)] を選択します。

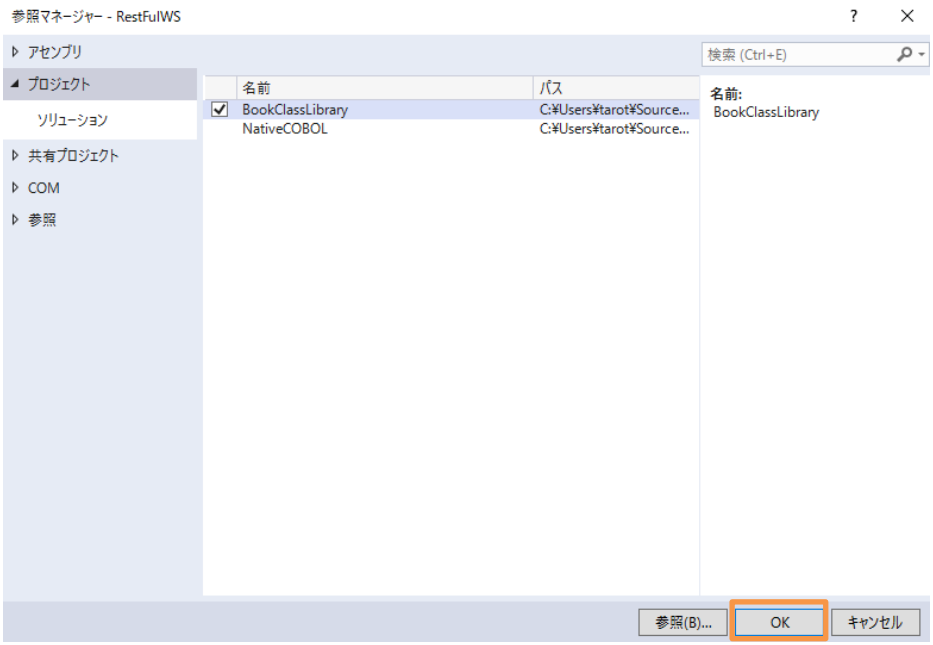


8) 「アセンブリ」 タブ配下の 「拡張」 タブを選択し、「Micro Focus Runtime」 と 「Micro Focus Runtime(Interop RuntimeServices)」 のチェックを行います。

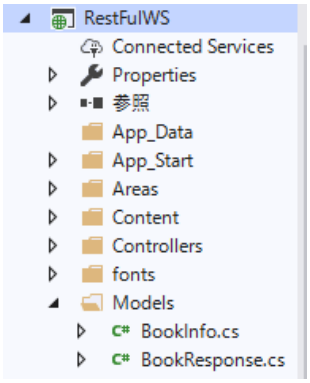


補足）
上記のように、複数のバージョンがリストアップされている場合は最新バージョンを選択してください。

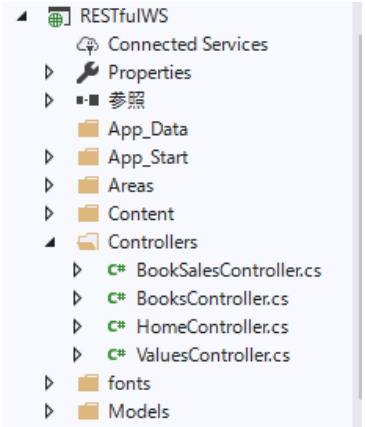
9) 「プロジェクト」タブを選択し、「BookClassLibrary」にチェックを行った後、[OK] をクリックします。



10) チュートリアルファイル解凍フォルダ¥WebService¥Models フォルダ配下の 2 ファイルを、RESTfulWS プロジェクト配下の Model フォルダにドラッグアンドドロップします。



11) チュートリアルファイル解凍フォルダ¥WebService¥Controllers フォルダ配下の 2 ファイルを RESTfulWS プロジェクト配下の Controllers フォルダにドラッグアンドドロップします。



12) BooksController.cs をダブルクリックして、COBOL 呼出し部の確認を行います。

```
public IHttpActionResult Get([FromUri]string id)
{
    System.Environment.SetEnvironmentVariable("BOOKINFO", System.Configuration.ConfigurationManager.AppSettings.Get("BOOKINFO"));

    BOOK book = new BOOK();
    LnkFunction lnkFunction = new LnkFunction();
    LnkFileStatus lnkFileStatus = new LnkFileStatus();
    LnkBDetails lnkBDetails = new LnkBDetails();

    RunUnit runUnit = new RunUnit();
    runUnit.Add(book);
    runUnit.Add(lnkFunction);
    runUnit.Add(lnkFileStatus);
    runUnit.Add(lnkBDetails);
    lnkFunction.LnkFunction = "1";
    lnkBDetails.LnkBStockno = id;
    book.BOOK(lnkFunction, lnkBDetails, lnkFileStatus);

    BookResponse bookResponse = new BookResponse();
    BookInfo outBookInfo = new BookInfo();
    outBookInfo.Title = lnkBDetails.LnkBTitle;
    outBookInfo.Genre = lnkBDetails.LnkBType;
    outBookInfo.Author = lnkBDetails.LnkBAuthor;
    outBookInfo.StockNo = lnkBDetails.LnkBStockno;
    outBookInfo.Retail = lnkBDetails.LnkBRetail;
    outBookInfo.Onhand = lnkBDetails.LnkBOnhand;
    outBookInfo.Sold = lnkBDetails.LnkBSold;
    bookResponse.bookInfo = outBookInfo;
    bookResponse.fileStatus = ((int)lnkFileStatus.LnkFileStatus.ElementAt(0)).ToString("X") + "/" + ((int)lnkFileStatus.LnkFileStatus.ElementAt(1)).ToString("X");
    runUnit.StopRun();

    return Ok(bookResponse);
}
```

.NET COBOL 機能により、従来の COBOL プログラムから PROGRAM-ID 名でクラス、および、メソッドが自動作成されます。今回の BOOK.cbl は PROGRAM-ID が “BOOK” であるため、BOOK クラスとなります。また、前手順で設定した ILSMARTLINKAGE コンパイラ指令により、LINKAGE SECTION に指定されていた LNK-FUNCTION, LNK-FILE-STATUS, LNK-B-DETAILS に対応するラッパークラスも合わせて作成されています。このラッパークラスを利用することで、 .NET 言語と COBOL のデータ型の差異を意識することなく、一般的な C# のコーディングで記述できていることが確認できます。

```
BOOK book = new BOOK();
LnkFunction lnkFunction = new LnkFunction();
LnkFileStatus lnkFileStatus = new LnkFileStatus();
LnkBDetails lnkBDetails = new LnkBDetails();

RunUnit runUnit = new RunUnit();
runUnit.Add(book);
runUnit.Add(lnkFunction);
runUnit.Add(lnkFileStatus);
runUnit.Add(lnkBDetails);
lnkFunction.LnkFunction = "1";
lnkBDetails.LnkBStockno = id;
book.BOOK(lnkFunction, lnkBDetails, lnkFileStatus);
```

補足）
多くの COBOL プログラムは、一般的にシングルスレッドでの動作を前提としています。しかし、Web アプリケーションは一般的にマルチスレッドのため、WORKING-STORAGE SECTION のようなプロセス共有資源のスレッド間衝突による問題が発生する可能性があります。 MicroFocus.COBOL.RuntimeServices.RunUnit は、これらの共有資源を各スレッドで独立して利用できるようにします。

13) アプリケーションの設定ファイルを修正します。RestfulWS プロジェクト配下の Web.config をダブルクリックします。

14) 以下の変更を行った後、保存します。

/configuration/appSettings 配下に、以下を追加

```
<add key="BOOKINFO" value="C:/DAT/BOOKINFO.DAT" />
```

```
<configuration>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0"/>
    <add key="webpages:Enabled" value="false"/>
    <add key="ClientValidationEnabled" value="true"/>
    <add key="UnobtrusiveJavaScriptEnabled" value="true"/>
    <add key="BOOKINFO" value="C:/DAT/BOOKINFO.DAT" />
  </appSettings>
```

/configuration/system.webServer 配下の以下の行をコメントアウト。

```
<remove name="OPTIONSVerbHandler"/>
```

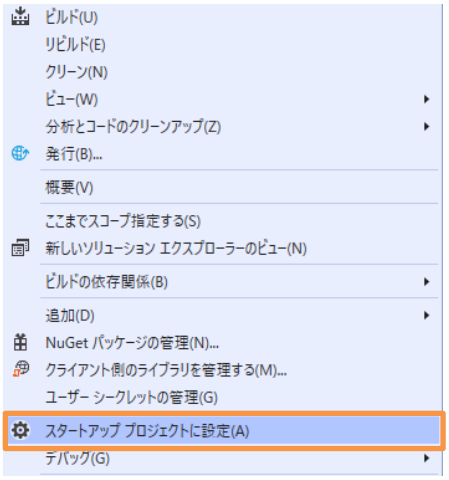
```
<configuration>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0"/>
    <add key="webpages:Enabled" value="false"/>
    <add key="ClientValidationEnabled" value="true"/>
    <add key="UnobtrusiveJavaScriptEnabled" value="true"/>
      <add key="BOOKINFO" value="C:/DAT/BOOKINFO.DAT" />
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.7.2"/>
    <httpRuntime targetFramework="4.7.2"/>
  </system.web>
  <system.webServer>
    <handlers>
      <remove name="ExtensionlessUrlHandler-Integrated-4.0"/>
        <!--
      <remove name="OPTIONSVerbHandler"/>
      -->
      <remove name="TRACEVerbHandler"/>
      <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*." verb="*" type="System.Web.Handlers.TransferRequestHandler"
        preCondition="integratedMode,runtimeVersionv4.0"/>
    </handlers>
```

/configuration/system.webServer 配下に、以下を追加。

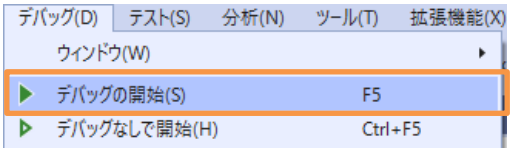
```
    <httpProtocol>
      <customHeaders>
        <add name="Access-Control-Allow-Origin" value="*" />
        <add name="Access-Control-Allow-Methods" value="*" />
        <add name="Access-Control-Allow-Headers" value="*"/>
      </customHeaders>
    </httpProtocol>
```

```
<configuration>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0"/>
    <add key="webpages:Enabled" value="false"/>
    <add key="ClientValidationEnabled" value="true"/>
    <add key="UnobtrusiveJavaScriptEnabled" value="true"/>
      <add key="BOOKINFO" value="C:/DAT/BOOKINFO.DAT" />
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.7.2"/>
    <httpRuntime targetFramework="4.7.2"/>
  </system.web>
  <system.webServer>
    <handlers>
      <remove name="ExtensionlessUrlHandler-Integrated-4.0"/>
        <!--
      <remove name="OPTIONSVerbHandler"/>
      -->
      <remove name="TRACEVerbHandler"/>
      <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*." verb="*" type="System.Web.Handlers.TransferRequestHandler"
        preCondition="integratedMode,runtimeVersionv4.0"/>
    </handlers>
    <httpProtocol>
        <customHeaders>
            <add name="Access-Control-Allow-Origin" value="*" />
            <add name="Access-Control-Allow-Methods" value="*" />
            <add name="Access-Control-Allow-Headers" value="*"/>
        </customHeaders>
    </httpProtocol>
  </system.webServer>
  <runtime>
```

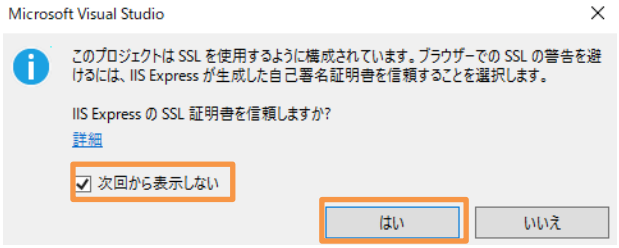
15) ソリューションエクスプローラーの RESTfulWS プロジェクト名を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[スタートアッププロジェクトに設定(A)] を選択します。



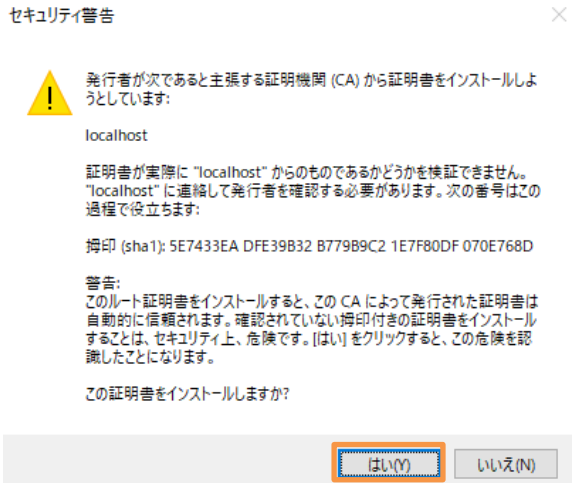
16) [デバッグ(D)] > [デバッグの開始(S)] を選択し、デバッグを行います。



以下のようなダイアログが表示された場合は、「次回から表示しない」にチェックを行い、[はい] をクリックします。



さらに、[はい(Y)] をクリックします。



17) インターネットエクスプローラーが立ち上がります。

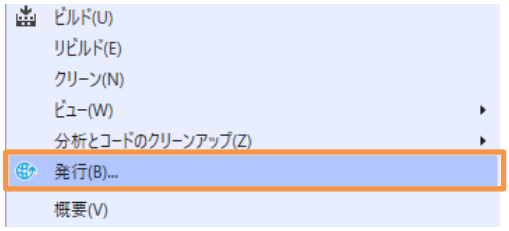
https://localhost:XXXXX/api/Books/1111 にアクセスします。

XXXXX にはポート番号を入力してください。ポート番号は、起動時の URL から確認できます。

上記のように、ストック番号： 1111 の情報が戻されることを確認して、インターネットエクスプローラーをクローズしてください。

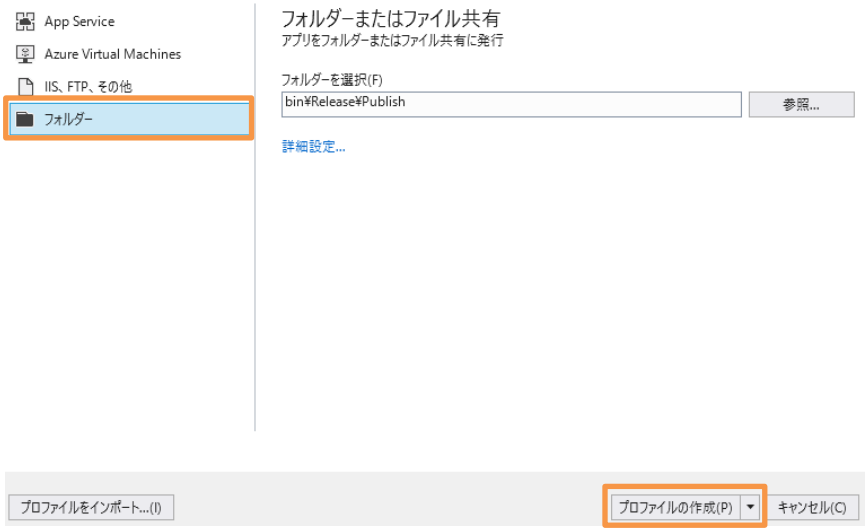
3.5. IIS サーバーへのサービス配置

1) RESTfulWS プロジェクト名を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[発行(B)] を選択します。



2) 「フォルダー」タブを選択し、[プロファイルの作成(P)] をクリックします。

発行先を選択



今回は出力先のフォルダを変更していないため、プロジェクトが保存されたフォルダ配下が出力先となります。デフォルトでは、以下のフォルダになります。

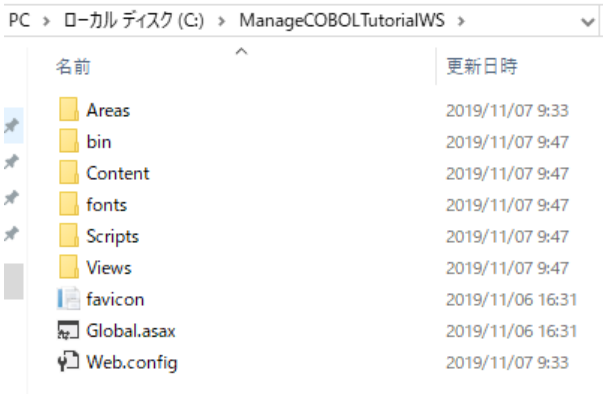
C:¥Users¥XXXXX¥Source¥Repos¥ManageCOBOLTutorial¥RestFulWS

(XXXXX にはログインユーザー名が入ります。)

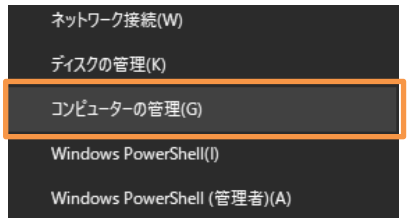
3) [発行(U)] をクリックします。

4) 前手順で出力されたリソース (bin¥Release¥Publish 配下のフォルダ・ファイル) を IIS サーバーに公開するため、以下のフォルダを作成し、コピーしてください。

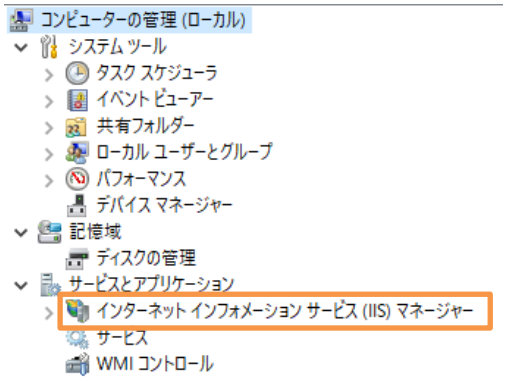
C:¥ManageCOBOLTutorialWS



5) Windows のスタートメニューの上で、右クリックにてコンテクストメニューを表示し、[コンピューターの管理(G)] を選択します。

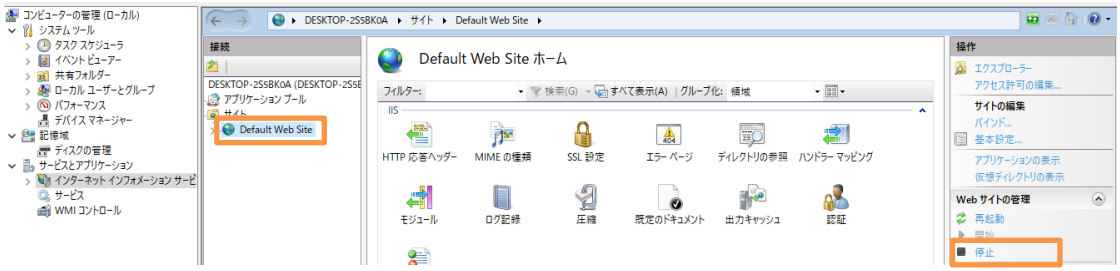


6) [サービスとアプリケーション] 配下の [インターネットインフォメーションサービス (IIS) マネージャー] をクリックします。

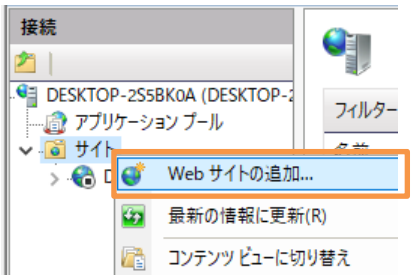


本項目が表示されていない方は、IIS がインストールされていません。4.1 の手順にてインストールを行ってください。

7) 本チュートリアル用に新規サイトを作成するため、[サイト] > [Default Web Site] を選択し、画面右側より [停止] をクリックします。



8) [サイト] を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[Web サイトの追加] をクリックします。



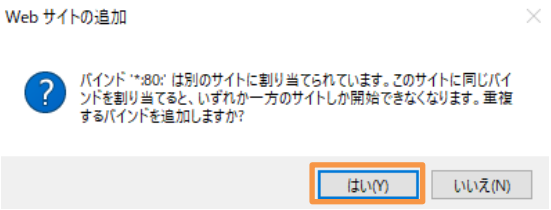
9) 以下の入力を行い、[OK] をクリックします。

サイト名： “tut”

物理パス： “C:¥ManageCOBOLTutorialWS”

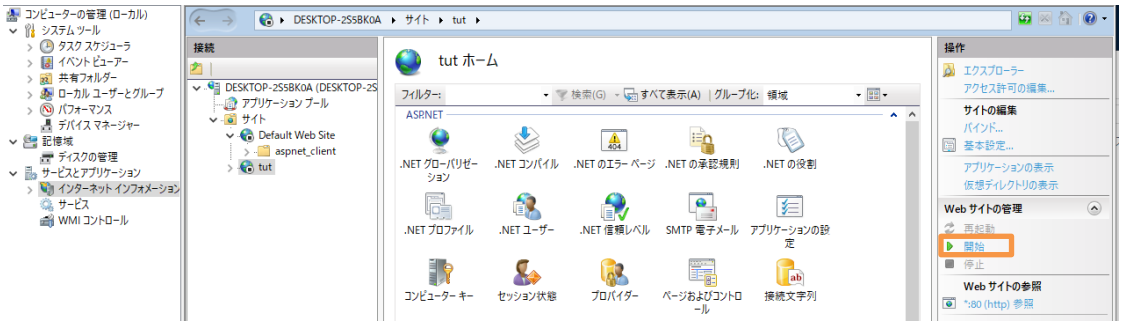
Web サイトを直ちに開始する： チェックをはずす

以下のダイアログには、[はい(Y)] をクリックします。



3.6. RESTful Web サービスの確認

1) 前手順で使用した “インターネット インフォメーションサービス (IIS) マネージャー” に戻り、”tut” サイトを選択し、画面右側より [開始] をクリックします。



2) Web ブラウザーを開き、以下の URL にアクセスします。

http://localhost/api/Books/1111

さきほどのデバッグ時と同様、書籍情報が JSON 形式で戻されているため、C# で作成された RESTful Web サービスが COBOL プログラムを正しく呼び出していることが分かります。

3) チュートリアルファイル解凍フォルダ配下の WebClient フォルダ配下の Search.html を Web ブラウザーで開いてください。

こちらは、今回作成した RESTful Web サービスを呼び出す Web 画面となります。さきほど確認したように REST API を介して JSON 形式でデータの送受信を行うモダンなシステムインターフェースとなっています。

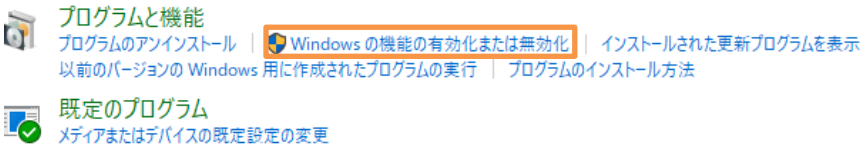
4. 補足

4.1. IIS サーバーのインストール方法

1) Windows でコントロールパネルを開きます。

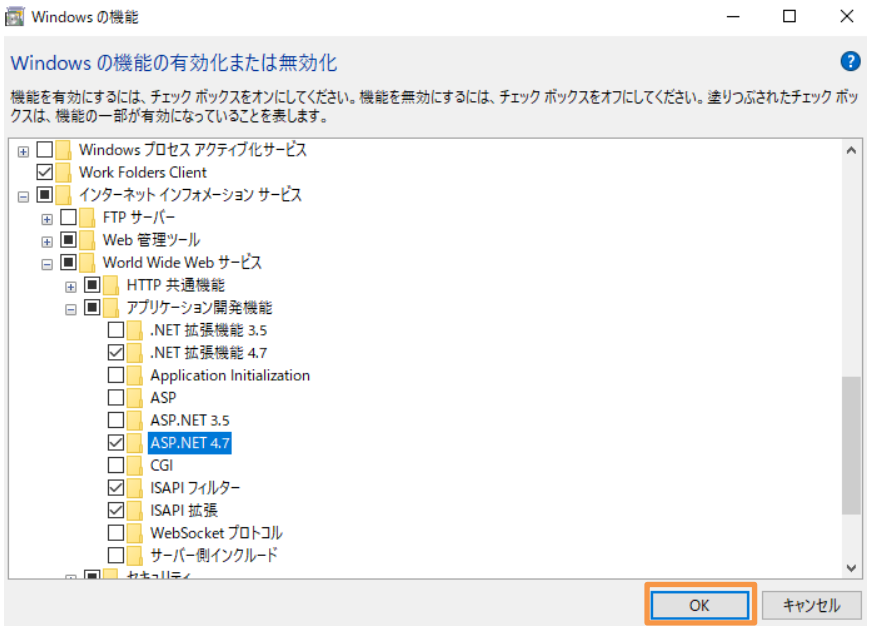
2) [プログラム] をクリックします。

3) [Windows の機能の有効化または無効化] をクリックします。



4) 以下のチェックを行い、 [OK] をクリックします。

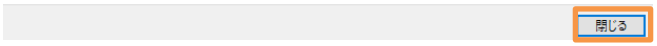
- 「インターネット インフォメーションサービス」配下の「Web 管理ツール」
- 「インターネット インフォメーションサービス」配下の「World Wide Web サービス」
- 「インターネット インフォメーションサービス」>「World Wirde Web サービス」>「アプリケーション開発機能」配下の「ASP.NET 4.7」[1]



[1] チェックを行うと、他の項目に対しても自動的にチェックが行われますが、そのままにしてください。

インストール完了後、[閉じる] をクリックします。

必要な変更が完了しました。



WHAT'S NEXT

- 本チュートリアルで学習した技術の詳細については製品マニュアルをご参照ください。

免責事項

ここで紹介したソースコードは、機能説明のためのサンプルであり、製品の一部ではございません。ソースコードが実際に動作するか、御社業務に適合するかなどに関しまして、一切の保証はございません。 ソースコード、説明、その他すべてについて、無謬性は保障されません。
ここで紹介するソースコードの一部、もしくは全部について、弊社に断りなく、御社の内部に組み込み、そのままご利用頂いても構いません。
本ソースコードの一部もしくは全部を二次的著作物に対して引用する場合、著作権法の精神に基づき、適切な扱いを行ってください。